A

FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.

1300 I STREET, N. W.
WASHINGTON, DC 20005-3315

202-408-4000
FACSIMILE 202-408-4400

AVENUE LOUISE 326, BOX 37
1050 BRUSSELS, BELGIUM
TELEPHONE 011-322-646-0353
FACSIMILE 011-322-646-2135

TORANOMON NO. 45 MORI BUILDING
THIRD FLOOR
1-5, TORANOMON 5-CHOME
MINATO-KU, TOKYO 105, JAPAN
TELEPHONE 011-813-3431-6943
FACSIMILE 011-813-3431-6945



WRITER'S DIRECT DIAL NUMBER:

(202) 408-4023

October 15, 1997

ATTORNEY DOCKET NO. 06502.0063-00000

**Box PATENT APPLICATION**
**Assistant Commissioner for Patents**
**Washington, D.C. 20231**

Re: New U.S. Patent Application
Title: REMOTE OBJECT ACTIVATION
IN A DISTRIBUTED SYSTEM
Inventors: Ann M. WOLLRATH et al.

Sir:

We enclose the following papers for filing in the United States Patent and Trademark Office in connection with the above patent application.

1. Application - 18 pages, including 6 independent claims and 27 claims total.

2. Drawings - 4 sheets of formal drawings.

3. A check for $1,190.00 representing a $790.00 filing fee and $400.00 for additional claims.

This application is being filed under the provisions of 37 C.F.R. § 1.53(d). Applicants await notification from the Patent and Trademark Office of the time set for filing the Declaration.

Please accord this application a serial number and filing date.

The Commissioner is hereby authorized to charge any additional filing fees due and any other fees due under 37 C.F.R. § 1.16 or § 1.17 during the pendency of this application to our Deposit Account No. 06-0916.

Respectfully submitted,

FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, L.L.P.

By: ____________________
Jeffrey A. Berkowitz
Reg. No. 36,743

JAB\BEL\tas
Enclosures

United States Patent Application

of

Ann M. Wollrath, Peter C. Jones, James H. Waldo, Robert W. Schiefler

for

REMOTE OBJECT ACTIVATION IN A DISTRIBUTED SYSTEM

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

## BACKGROUND OF THE INVENTION

The present invention relates generally to distributed computer systems, and more specifically, to managing and activating objects in distributed computer systems.

A distributed computer system is a network of computer processors that although geographically separated, are linked together functionally. It is often desirable to execute a computer program, or portions of a computer program, simultaneously across several computer processors in the distributed system. In such an environment, protocols coordinating the various portions of the program(s) are necessary.

Distributed computing systems executing object-oriented programming models are known. Essentially, in these systems, programs are written as an aggregation of objects, each of which may reside on and be executed on a different computer in the distributed system.

Typically, in an object-oriented distributed system, a local computer system, called the client, may access objects on remote computer systems. If the objects to be accessed on the remote computer system take up processor resources, i.e., if they consume physical or virtual memory and have a thread of control, they are said to be "active." Examples of such active objects include running programs or objects that are part of active programs. Such objects are always taking up resources from the physical machine, even when they are not doing active work on behalf of themselves or at the request of some other object.

A "passive" object, on the other hand, refers to a presently non-active object on the remote computer. If a passive object is "activatable," it may, at the request of the client computer system, be brought into an active state. Objects may be passive simply because they have never

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

been instantiated. Alternatively, to save system resources, active objects may be de-activated and become passive. In particular, for active objects that have become quiescent, it may be advantageous for the computer to save the state information of the object to a stable storage medium, such as a magnetic disk, and release any memory or threads of control associated with the object. The de-activated object does not take up physical or virtual memory and is not associated with a control thread, although it continues to exist and may be made active when called.

One known distributed system capable of activating objects is the object management groups Common Object Request Broker Architecture (CORBA) system. In the CORBA system, remote objects are always considered by the client to be potentially passive, and thus activatable, regardless of whether the object is actually active or passive. Additionally, although some objects at a remote system may be similar to one another, and capable of benefiting from a sharing of common resources, CORBA does not provide for the associating of similar objects.

There is, therefore, a need for a distributed system object management architecture that solves the above mentioned limitations found in the prior art.

SUMMARY OF THE INVENTION

Objects and advantages of the invention will be set forth in part in the description which follows, and in part will be obvious from the description, or may be learned by practice of the invention. The objects and advantages of the invention will be realized and attained by means of the elements and combinations particularly pointed out in the appended claims.

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
1300 I STREET, N.W.
WASHINGTON, D.C. 20005
202-408-4000

To achieve the objects and in accordance with the purpose of the invention, as embodied and broadly described herein, a first aspect of the present invention includes a method of calling a remote object by a process comprising the steps of: (1) calling the remote object directly using a first address in a faulting remote reference to the remote object when the reference refers to an active instance of the remote object; and (2) calling an activator object using a second address in the faulting remote reference to perform activation of the remote object when the reference to the remote object does not refer to an active instance of the remote object. In an alternative aspect, a computer readable medium contains instructions for performing similar steps.

A second method consistent with the present invention includes a method of handling an object call at a remote site for a remote object, the method comprises the steps of: (1) determining whether a first predefined group of objects corresponding to the called remote object is active; (2) activating the remote object within the first group when the determining step determines that the first group is active; and (3) creating a second group of objects and activating the remote object within the second group when the determining step determines that the first group is not active. As with the first method, an alternative aspect includes a computer readable medium containing instructions for performing similar steps.

Still further, a distributed computer system consistent with the present invention includes a plurality of elements, including first and second computers. The second computer, in particular, receives requests for remote objects from the first computer and executes an object activator performing the steps of: (1) determining whether a first predefined group of objects corresponding to the requested remote object is active; (2) activating the requested remote object

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT
& DUNNER, L. L. P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

within the first group of objects when the determining step determines that the first group of objects is active; and (3) creating a second group of objects and activating the requested remote object within the second group of objects when the determining step determines that the first group of objects is not active.

## BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate several embodiments consistent with this invention and, together with the description, help explain the principles of the invention. In the drawings,

Fig. 1 is a high-level block diagram illustrating interaction of hardware and software components in an exemplary distributed computer system consistent with the present invention;

Fig. 2 is a block diagram illustrating software entities located on a local computer; and

Fig. 3 is a block diagram illustrating software entities located on a remote host computer; and

Fig. 4 is a flow chart illustrating steps consistent with the present invention.

## DESCRIPTION OF THE PREFERRED EMBODIMENT

A distributed computer system and related methods consistent with the present invention are described herein. The distributed computer system uses a single interface at the client site to handle calls to call both active and activatable remote objects. Further, remote objects are

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L. L. P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

aggregated into common groups of objects, thereby providing greater security between objects of disparate groups and efficiency between related objects of the same group.

Wherever possible, the same reference numbers will be used throughout the drawings to refer to the same or like parts.

Fig. 1 is a high-level block diagram illustrating interaction of hardware and software components in an exemplary distributed computer system. Computer 102 includes a computer hardware/processing section 107 executing programs from memory section 103. Memory 103 is preferably a random access memory, but may additionally or alternatively include other storage media such as magnetic or optical disks.

Memory 103 stores one or more computer procedures 104a, 104b, and 104c, such as, for example, a computer program, thread, or object. Computer threads 104a and 104b are programs comprised of Java bytecodes executing through a Java virtual machine 105. The virtual machine is itself a process that when executed on computer 102, translates threads 104a and 104b into computer instructions native to computer hardware 107. In this manner, virtual machine 105 acts as an interpreter for computer hardware 107. In contrast to threads 104a and 104b, program 104c uses instructions native to computer hardware 107, and thus does not require virtual machine 105.

Computer 102 is connected via network 120 to computer 112. Computer 112 includes components similar to those of computer 102, and will therefore not be described further. Although the simple network described above includes only two computers, networks of many

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

computers, or even networks of many thousands of computers, such as the Internet, may be used to implement the concepts of the present invention.

Throughout the remainder of this disclosure, computer system 102 will be described as the requester of remote objects. Computer system 112 executes the remote objects and returns results to computer 102. Although not explicitly shown, a plurality of computer systems 112 may execute multiple objects for a single host computer 102.

Fig. 2 is a block diagram illustrating software entities located on computer 102.

Process 202 is a program active on computer 102, such as process 104 in Fig. 1. As shown, process 202 includes a plurality of bytecodes that may be translated from instructions written in the Java programming language, including instruction 203, which is an invocation to a method residing in an object on remote computer 112. The method invocation is preferably defined to be handled by local proxy object 205, which functions as an interface for remote object calls from computer 102 and hides the remote calling protocol from the invocating process.

Proxy 205 may assume one of multiple implementations depending on the status of the object being referenced; such as whether the object is active or activatable (i.e., presently passive). When called by process 202, proxy 205 packages the call using the appropriate implementation and forwards it to remote computer 112. Results received from the remote computer, such as results from the method invocation, are passed back through proxy 205 to process 202.

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L. L. P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

As described in more detail below, proxy 205 enables process 202 to make a single method invocation for both active and activatable objects. In other words, process 202 is not required to monitor whether a remote object is active or activatable.

Activation of remote objects by proxy 205 is implemented through an object reference known as a faulting remote reference, illustrated by reference 210. For each remote object, faulting remote reference 210 is used to "fault in" the object's reference upon the first method invocation to the object. Faulting remote reference 210 includes a persistent handle (an activation identifier) 211 and a transient remote reference 212. Both persistent handle 211 and transient remote reference 212 are obtained from the remote computer corresponding to the remote object, and contain address information for contacting the remote computer, such as the appropriate network address and port number, and address information more specific to the remote object being referred. Persistent handle 211 is the more general reference and references an activator entity (described in more detail below) at the remote host. Reference 212 is the actual "live" reference to the active remote object, and is used to contact the remote object directly.

In operation, upon invocation of a method requiring a remote object, proxy 205 checks reference 210. A null value in "live" reference 212 indicates that the remote object may become passive (i.e., it is not an active-only object), and proxy 205 uses activation identifier 211 to contact an activator entity at the remote site. If reference 212 is not null it will point directly to the remote object. This indicates an active remote object, which proxy 205 contacts directly.

Fig. 3 is a block diagram illustrating software entities located on host computer 112. As mentioned previously, host computer 112 is contacted by the client using either the activation identifier reference 211 or "live" reference 212. Activation identifier reference 211 references object activator 302, which supervises object activation on the host. Activator 302 functions as: (1) a database that maps activation identifiers 211 to the information necessary to activate an object (e.g., the object's class, the location-- a URL-- from where the class can be loaded, specific data the object may need to bootstrap, etc.); (2) a database for tracking the current mapping of activation identifiers to active objects; and (3) a manager of Java virtual machines.

Activatable objects are defined by the designer to exist as a member of a group of objects, such as group 305. The designer preferably assigns objects to particular groups so that objects within a group are designed to interact with one another. For example, objects within a group should have a relationship of mutual trust strong enough to allow them all to run within a single Java virtual machine. Once assigned to a group, objects stay within that group.

Activation entity 304 manages object group 305. In particular, activation entity 304 activates passive objects and creates objects pursuant to requests from object activator 302, and returns a reference to the corresponding activated object to object activator 302. To activate a quiescent object within group 305, activation entity 304 allocates the appropriate operating system resources (memory, process, or thread allocation) and starts up the object. After activating an object, activation entity 304 passes information to object activator 302 describing the way in which the object is to be reached for further communication. Object activator 302 may then forward this information to proxy 205, which appropriately updates faulting remote

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
1300 I STREET, N.W.
WASHINGTON, D.C. 20005
202-408-4000

reference 210. If an object later de-activates, or is de-activated, object activator 302 similarly communicates with proxy 205 to update the faulting remote reference.

Preferably, one activation entity 304 exists per each active Java virtual machine.

Fig. 4 is a flow chart further illustrating steps consistent with the present invention. Upon invocation of a remote object by computer 102, proxy 205 determines if the transient remote reference (the "live" reference) 212 at computer 102 is present, i.e., if it is not null, and proxy 205 contacts the active remote object (steps 402, 404). Otherwise, proxy 205 uses persistent handle 211 within reference 210 to contact object activator 302 (steps 402, 406). Object activator 302 uses information within reference 210 to determine if an object group corresponds to the invocated object (step 408). If an appropriate group is already active, an activation request for the called object is forwarded to the appropriate activation entity (steps 408, 410), and the object is activated (step 411). Otherwise, the activator object first creates a new virtual machine and a new activation entity (steps 408, 412), and then forwards the activation request to the newly created activation entity, (step 414), at which point the object is activated (step 415). In response to forwarding an activation request to an activation entity, the object activator will usually receive an updated network address and port number, which it forwards to proxy object 205 (step 416).

As described above, object groups such as group 305 form the basic unit of object activation. Object activator 302 and activation entities 304 manage the object groups, such that if a group has not been activated then a call to any object of an object group will cause the activation of that object group and the called object in a new Java virtual machine.

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L. L. P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

Clustering objects within an object group on a single Java virtual machine allows related objects to share an address space, which in turn allows close communication between the objects. Objects in different groups, on the other hand, are in different Java virtual machines and thus have a much stronger security separation, ensuring that those objects will not interfere, either intentionally or unintentionally, with each other.

Further, a single interface is seen by clients attempting to call remote objects. The interface has multiple implementations depending on the status of the object being referenced, allowing for transparent mixing of active and passive (i.e., activatable) objects in the same system, supporting both without requiring that the clients of those objects have any knowledge of whether or not the object is activatable. This interface provides the client the ability to make any calls that are supported by the remote object through a faulting remote reference.

It will be apparent to those skilled in the art that various modifications and variations can be made to the concepts of the present invention and in its construction without departing from the scope or spirit of the invention. Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. It is intended that the specification and examples be considered as exemplary only, with a true scope and spirit of the invention being indicated by the following claims.

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L. L. P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

WHAT IS CLAIMED IS:

1. A method of calling a remote object by a process comprising the steps of:

calling the remote object using a first address in a faulting remote reference to the remote object when the reference refers to an active instance of the remote object; and

calling an activator object using a second address in the faulting remote reference to perform activation of the remote object when the reference to the remote object refers to a null instance of the remote object.

2. The method of claim 1, further including the step of accessing an interface to call the remote object such that the steps of calling the remote object directly using the first address and calling an activator object using the second address are performed transparently to the process.

3. The method of claim 1, further including the step of updating the faulting remote reference when a new version of the faulting remote reference is received from a computer associated with the remote object.

4. A computer readable medium containing instructions executable on a first computer for calling a remote object located at a second computer, the instructions causing the first computer to:

access the remote object directly at the second computer using a first address in a faulting remote reference to the remote object when the reference refers to an active instance of the remote object; and

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L. L. P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

access an activator object at the second computer using a second address in the faulting remote reference to activate the remote object when the reference to the remote object refers to a null instance of the remote object.

5. The computer readable medium of claim 4, further including instructions causing the first computer to access an interface to access the remote object such that the steps of accessing the remote object directly using the first address and accessing an activator object using a second address are performed transparently to a process initiating calling of the remote object.

6. The computer readable medium of claim 4, further including instructions causing the first computer to update the faulting remote reference when a new version of the faulting remote reference is received from the second computer.

7. A method of handling an object call at a remote site for a remote object, the method comprising the steps of:

receiving a call to activate the remote object;

determining whether a first predefined group of objects corresponding to the called remote object is active;

activating the remote object within the first group when the determining step determines that the first group is active; and

creating a second group of objects and activating the remote object within the second group when the determining step determines that the first group is not active.

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

8. The method of claim 7, wherein the step of activating the object within the first group further includes the step of activating the object within an address space of previous objects activated within the first group.

9. The method of claim 7, wherein the step of activating the remote object within the first group further includes the step of activating the object within a same Java virtual machine as previous objects activated within the first group.

10. The method of claim 7, wherein the step of creating the second group includes spawning a virtual machine to interpret the second group.

11. The method of claim 10, further including the step of returning results of the activated remote object.

12. A computer readable medium containing instructions executable on a remote computer in a network of distributed computers for handling an object call at the remote computer for a remote object, the instructions causing the computer to:

determine whether a first predefined group of objects corresponding to the called remote object is active;

activating the remote object within the first group when the determining step determines that the first group is active; and

creating a second group and activating the remote object within the second group when the determining step determines that the first group is not active.

13. The computer readable medium of claim 12, wherein the step of activating the remote object within the first group further includes further including instructions for performing

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

the step of activating the object within an address space of previous objects activated within the first group.

14. The computer readable medium of claim 12, wherein the step of activating the remote object within the first group further includes instructions for performing the step of activating the object within a same Java virtual machine as previous objects activated within the first group.

15. The computer readable medium of claim 12, wherein the step of creating the second group further includes instructions for spawning a virtual machine to interpret the second group.

16. The computer readable medium of claim 12, further including instructions for performing the step of returning results of the activated remote object.

17. A distributed computer network comprising:

a first computer executing a proxy object called by a process and instantiated as one of a plurality of different implementations depending on the process call; and

a second computer receiving requests for remote objects from the first computer and executing an object activator performing the steps of: (1) determining whether a first predefined group of objects corresponding to the requested remote object is active; (2) activating the requested remote object within the first group of objects when the determining step determines that the first group of objects is active; and (3) creating a second group of objects and activating the requested remote object within the second group of objects when the determining step determines that the first group of objects is not active.

18. The computer system of claim 17, wherein the plurality of different implementations of the proxy object form an interface such that details of the remote object call are hidden from the calling process.

19. The computer system of claim 17, wherein the proxy object further includes means for calling the remote object directly using a first address in a faulting remote reference to the remote object when the reference refers to an active instance of the remote object and means for calling the activator object using a second address in the faulting remote reference when the reference refers to a null instance of the remote object.

20. The computer system of claim 17, wherein the proxy object is a processes executing on a virtual machine.

21. The computer system of claim 17, wherein the object activator is a processes executing on a virtual machine.

22. A method of calling a remote object comprising the steps of:

receiving a request to access the remote object;

determining whether the remote object is active; and

accessing the remote object based on the results of the determination.

23. The method of claim 22, wherein the determining step includes the substep of maintaining a faulting remote reference to the remote object.

24. The method of claim 22, wherein the determining step includes the substep of maintaining a faulting remote reference to the remote object, and when the faulting remote

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L. L. P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

reference indicates that the remote object is active, the accessing step includes the substep of directly contacting the remote object.

25. The method of claim 22, wherein the determining step includes the substep of maintaining a faulting remote reference to the remote object, and when the faulting remote reference indicates that the remote object is not active, the accessing step includes the substep of instantiating the remote object.

26. The method of claim 25, wherein the instantiating step includes the substep of determining whether a virtual machine for the remote object is active.

27. The method of claim 26, wherein when the a virtual machine for the remote object is not active, the instantiating step further includes the step of spawning a new virtual machine for the remote object.



LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

## ABSTRACT

A distributed computer system uses a single interface at the client site to handle calls to call both active and passive remote objects. Accordingly, the calling process does not need to be aware of distinctions between active and passive objects. Further, remote objects are aggregated into common groups of objects, thereby providing greater security between objects of disparate groups and efficiency between related objects of the same group. Preferably, different groups are run on different Java virtual machines.

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

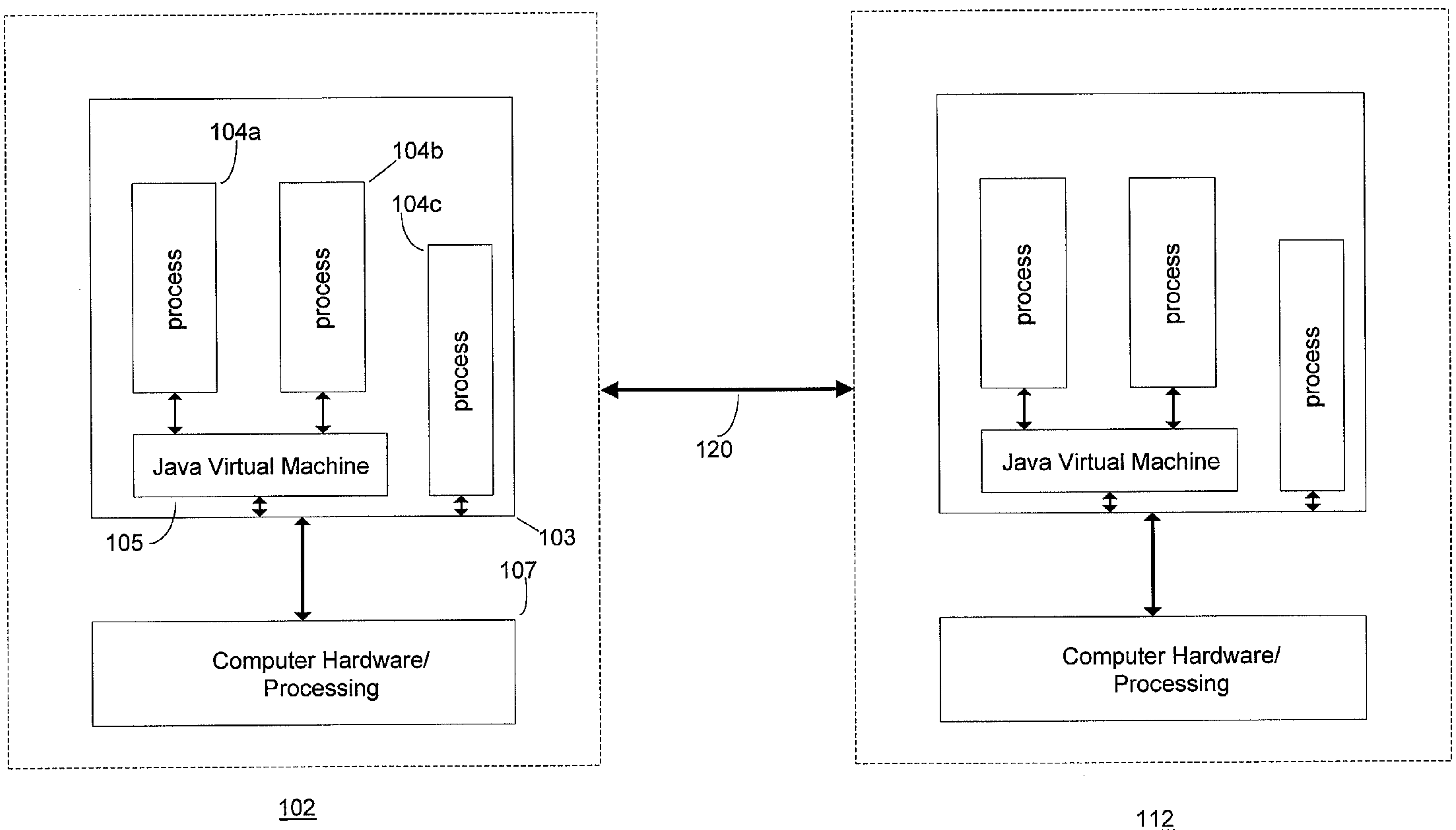
104a
104b
104c
process
process
process
Java Virtual Machine
105
103
107
Computer Hardware/
Processing
102
120
process
process
process
Java Virtual Machine
Computer Hardware/
Processing
112

Fig. 1

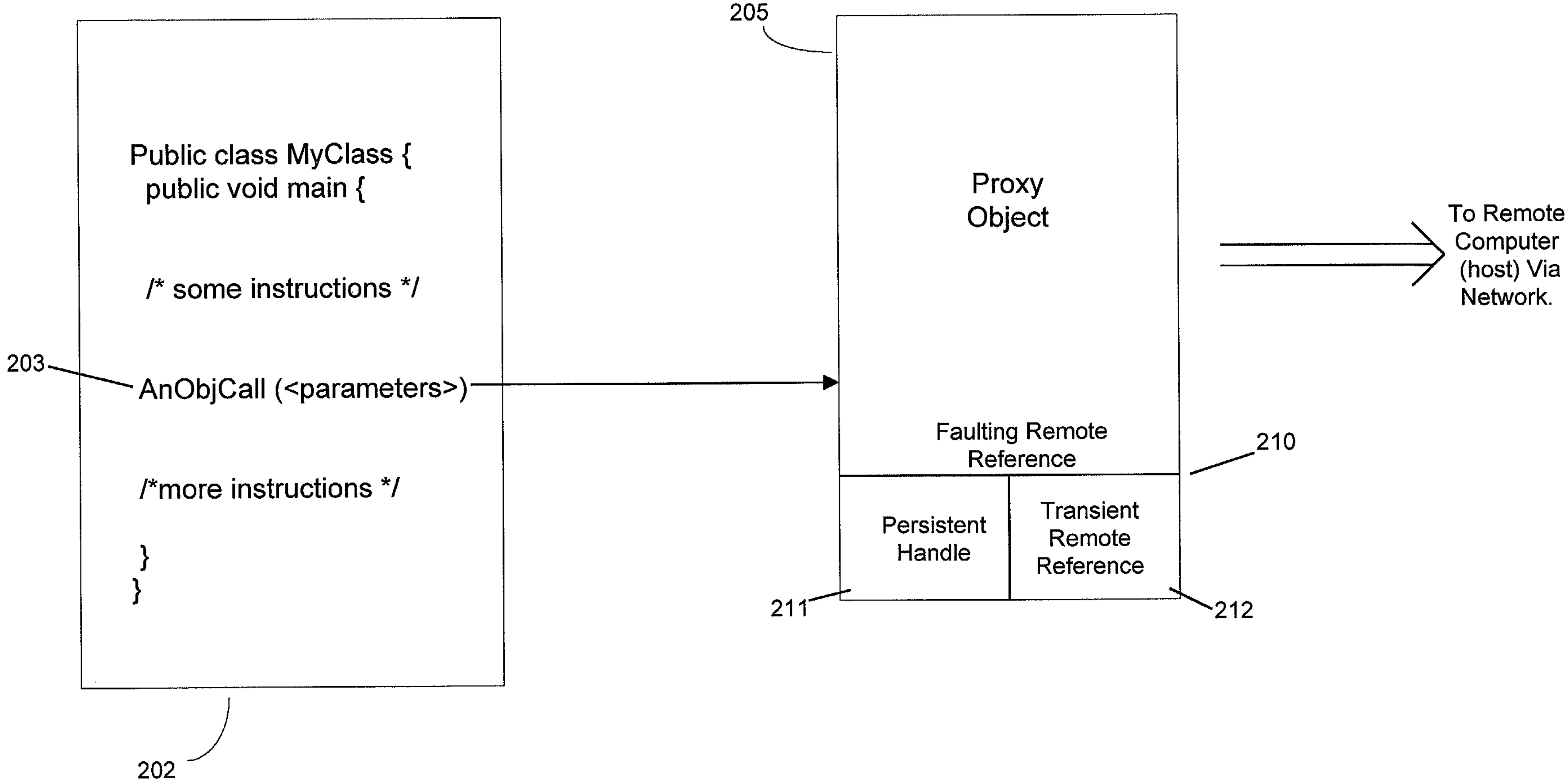
Public class MyClass {
public void main {
/* some instructions */
AnObjCall (<parameters>)
/*more instructions */
}
}
202
203
205
Proxy
Object
Faulting Remote
Reference
Persistent
Handle
Transient
Remote
Reference
210
211
212
To Remote
Computer
(host) Via
Network.

Fig. 2

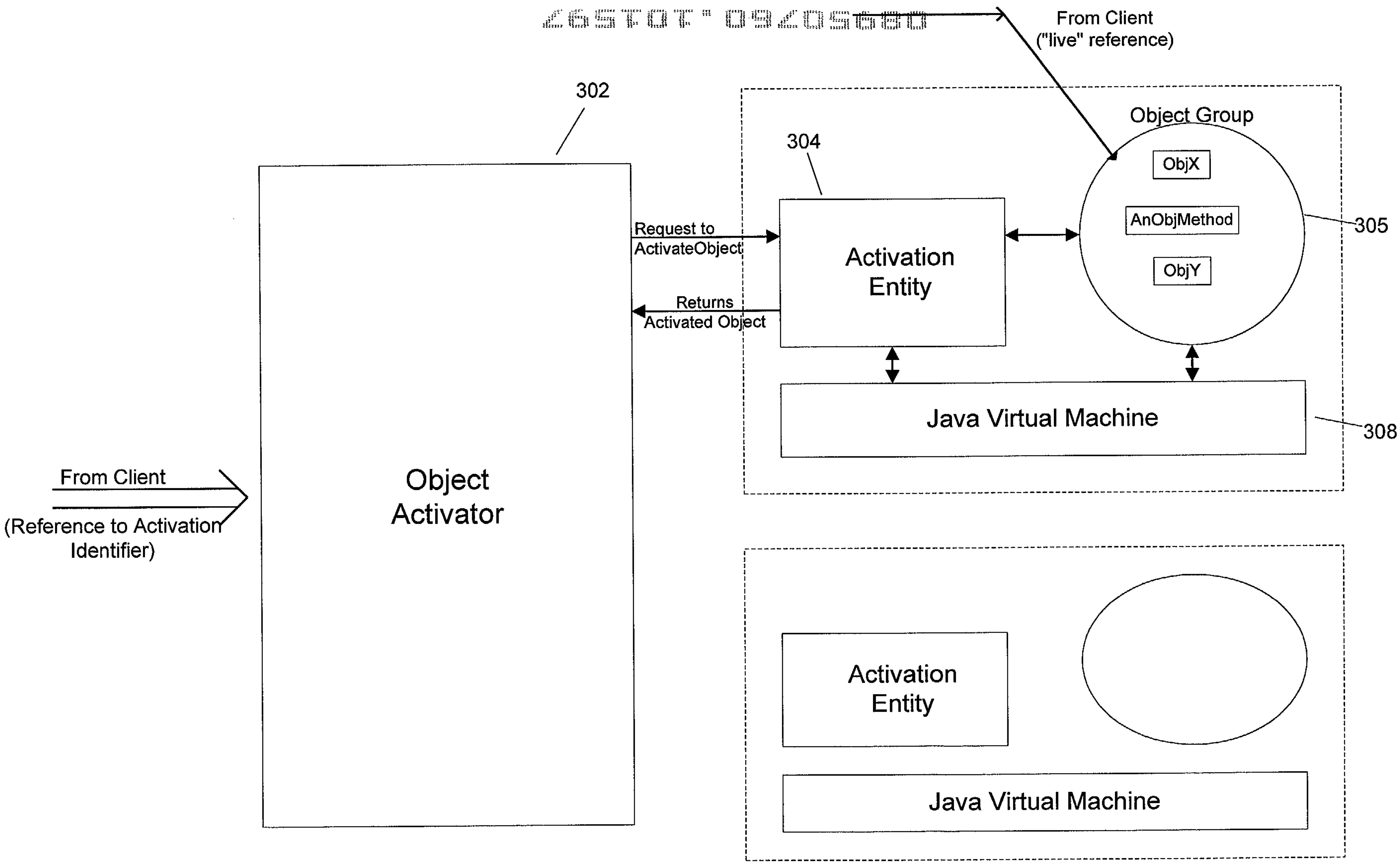
From Client
("live" reference)
302
304
Object Group
ObjX
AnObjMethod
ObjY
305
Request to
ActivateObject
Returns
Activated Object
Activation
Entity
Java Virtual Machine
308
From Client
(Reference to Activation
Identifier)
Object
Activator
Activation
Entity
Java Virtual Machine

Fig. 3

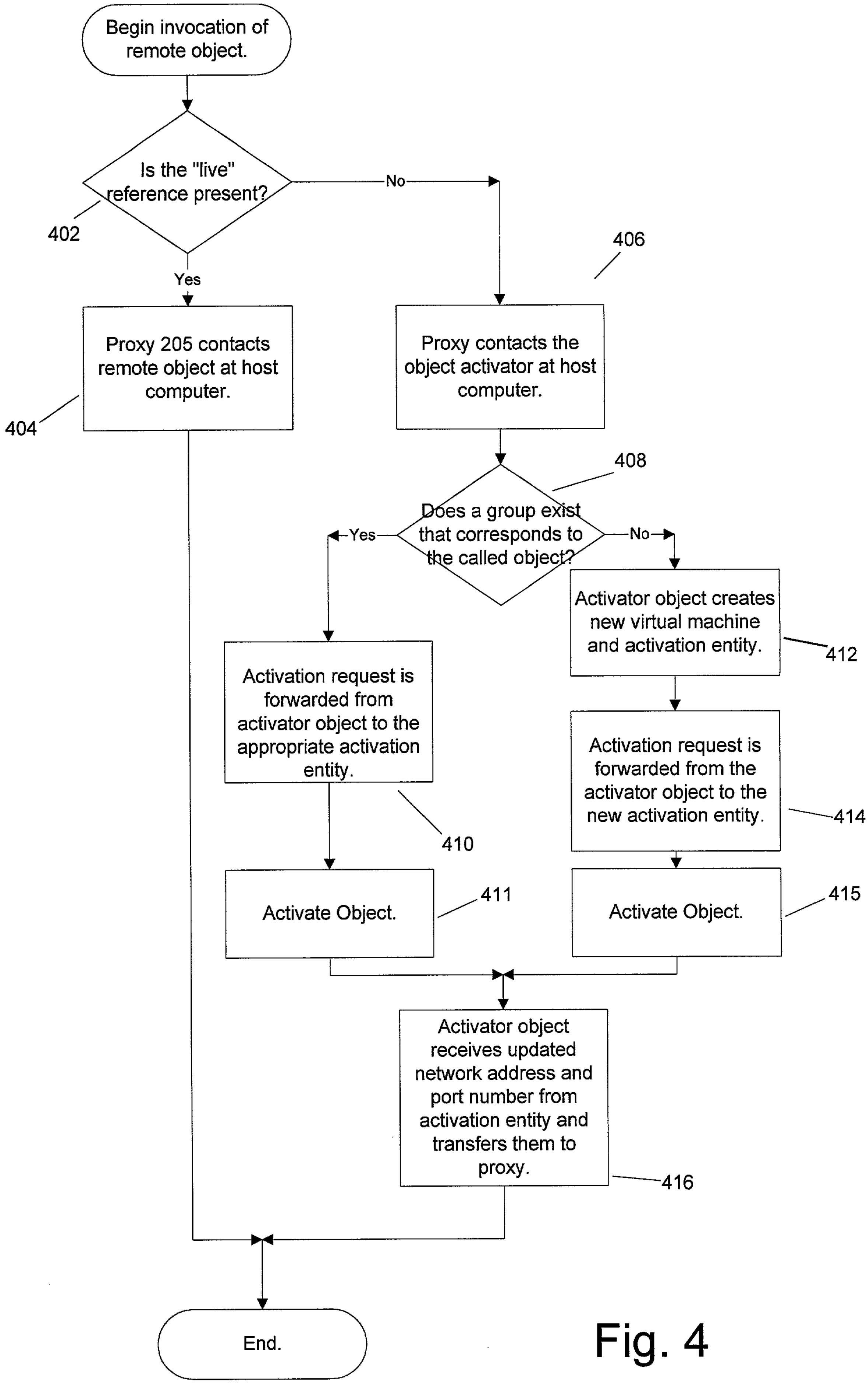
Begin invocation of remote object.
Is the "live" reference present?
402
No
Yes
406
Proxy 205 contacts remote object at host computer.
404
Proxy contacts the object activator at host computer.
408
Does a group exist that corresponds to the called object?
Yes
No
Activator object creates new virtual machine and activation entity.
412
Activation request is forwarded from activator object to the appropriate activation entity.
Activation request is forwarded from the activator object to the new activation entity.
414
410
Activate Object.
411
Activate Object.
415
Activator object receives updated network address and port number from activation entity and transfers them to proxy.
416
End.

Fig. 4